FORMATO DE GUÍA DE PRÁCTICA DE LABORATORIO / TALLERES / CENTROS DE SIMULACIÓN – PARA DOCENTES

CARRERA: Computación		**ASIGNATURA**: Programación Orientada a Objetos
NRO. PRÁCTICA:	1	**TÍTULO PRÁCTICA**: MVC.

OBJETIVO (Colocar el o los objetivos que se alcanzarán al desarrollar la práctica):

- Generar un sistema basado en MVC del siguiente enunciado.
-

INSTRUCCIONES (Detallar las instrucciones que se dará al estudiante):

1. Desarrollar un diagrama de clases que permita gestionar PROYECTOS.

2. A continuación, las especificaciones del **Diagrama de clases**:

   - Cada proyecto tiene un código y un nombre. Un proyecto tiene uno y solo un jefe de proyecto y un jefe de proyecto sólo puede estar involucrado en un proyecto o en ninguno.
   - De cada jefe de proyecto se desean recoger sus datos personales (código, nombre, dirección y teléfono). Un jefe de proyecto se identifica por un código. No hay dos nombres de jefe de proyecto con el mismo nombre.
   - Un proyecto se compone de una serie de planos, pero éstos se quieren guardar de modo independiente al proyecto. Es decir, si en un momento dado se dejara de trabajar en un proyecto, se desea mantener la información de los planos asociados.
   - De los planos se desea guardar su número de identificación, la fecha de entrega, los arquitectos que trabajan en él y un dibujo del plano general con información acerca del número de figuras que contiene.
   - Los planos tienen figuras. De cada figura se desea conocer, el identificador, el nombre, el color, el área y el perímetro. Además, de los polígonos se desea conocer el número de líneas que tienen, además de las líneas que lo forman. El perímetro se desea que sea un método diferido; el área se desea implementarlo como genérico para cualquier tipo de figura, pero además se desea un método específico para el cálculo del perímetro de los polígonos.
   - De cada línea que forma parte de un polígono se desea conocer el punto de origen y el de fin (según sus coordenadas, X e Y), así como la longitud. Cada línea tiene un identificador que permite diferenciarlo del resto. La longitud de la línea se puede calcular a partir de sus puntos origen y final.

5. Criterios de valoración:

Criterio	Puntaje
MVC	
• MVC	3
• Usabilidad	1
• Buenas prácticas en el nombre de clases y atributos	1
• Total	5

6. Consideraciones:

- La prueba se realizara individual.
- Tienen hasta 20/06/2021 23:55 para terminar a partir de la explicación.
- Generar un archivo **.pdf** el nombre debe tener el siguiente formato ApellidoNombre con las capturas del funcionamiento del sistema.
- Subir el codigo fuente y el archivo pdf al Git Perasonal con el nombre de la caperta PruebaPractica2.

ACTIVIDADES POR DESARROLLAR

(Anotar las actividades que deberá seguir el estudiante para el cumplimiento de la práctica)

1. Realizar el sistema empleando MVC para la gestión de datos y las buenas practicas de programación.

El diagrama de clases debe cumplir con las buenas prácticas indicadas en clase.

RESULTADO(S) OBTENIDO(S):
Escribir los resultados obtenidos con la realización de la práctica.

- Construye sistemas transaccionales utilizando MVC.

CONCLUSIONES:

- Cada estudiante podrá desarrollar un diagrama de clases identificando la información primordial en base a un problema.

RECOMENDACIONES:

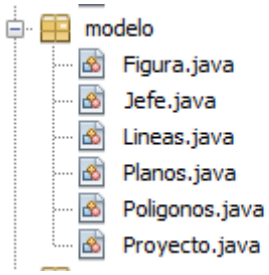
- Revisar el material de los ejercicios.
- Usar Dia.

Docente / Técnico Docente*: Ing. *Diego Quisi

***Firma*: _____**

Modelo

Se crea el modelo con todo lo que pide el enunciado, en cada uno de estos se realiza lo que es la creación de los controladores, getters y setters.



Modelo Jefe

```
public class Jefe {
    private String id;
    private String Nombre;
    private String Apellido;
    private long codigo;
    private String direccion;
    private String telefono;

    public Jefe(String Nombre, String Apellido, long codigo, String direccion, String telefono) {
        this.Nombre = Nombre;
        this.Apellido = Apellido;
        this.codigo = codigo;
        this.direccion = direccion;
        this.telefono = telefono;
    }

    public String getNombre() {
        return Nombre;
    }

    public void setNombre(String Nombre) {
        this.Nombre = Nombre;
    }

    public String getApellido() {
        return Apellido;
    }

    public void setApellido(String Apellido) {
        this.Apellido = Apellido;
    }

    public long getCodigo() {
        return codigo;
    }

    public void setCodigo(long codigo) {
        this.codigo = codigo;
    }
```

```java
public String getDireccion() {
    return direccion;
}

public void setDireccion(String direccion) {
    this.direccion = direccion;
}

public String getTelefono() {
    return telefono;
}

public void setTelefono(String telefono) {
    this.telefono = telefono;
}

@Override
public String toString() {
    return "Jefe{" + "Nombre=" + Nombre + ", Apellido=" + Apellido + ", codigo=" + codigo + ", direccion=" + direccion + ", telefono=" + telefono + '}';
}
```

Modelo Proyecto

```java
public class Proyecto {
    private long codigo;
    private String Nombre;

    public Proyecto(long codigo, String Nombre) {
        this.codigo = codigo;
        this.Nombre = Nombre;
    }

    public long getCodigo() {
        return codigo;
    }

    public void setCodigo(long codigo) {
        this.codigo = codigo;
    }

    public String getNombre() {
        return Nombre;
    }

    public void setNombre(String Nombre) {
        this.Nombre = Nombre;
    }

    @Override
    public String toString() {
        return "Proyecto{" + "codigo=" + codigo + ", Nombre=" + Nombre + '}';
    }
}
```

Modelo Planos

```java
public class Planos {
    private long Identificacion;
    private String Fecha;
    private int numArquitectos;
    private int numFiguras;

    public Planos(long Identificacion, String Fecha, int numArquitectos, int numFiguras) {
        this.Identificacion = Identificacion;
        this.Fecha = Fecha;
        this.numArquitectos = numArquitectos;
        this.numFiguras = numFiguras;
    }

    public long getIdentificacion() {
        return Identificacion;
    }

    public void setIdentificacion(long Identificacion) {
        this.Identificacion = Identificacion;
    }

    public String getFecha() {
        return Fecha;
    }

    public void setFecha(String Fecha) {
        this.Fecha = Fecha;
    }

    public int getNumArquitectos() {
        return numArquitectos;
    }

    public void setNumArquitectos(int arquitectos) {
        this.numArquitectos = numArquitectos;
    }

    public int getNumFiguras() {
        return numFiguras;
    }

    public void setNumFiguras(int figuras) {
        this.numFiguras = numFiguras;
    }

    @Override
    public String toString() {
        return "Planos{" + "Identificacion=" + Identificacion + ", Fecha=" + Fecha + ", numArquitectos=" + numArquitectos + ", numFiguras=" + numFiguras + '}';
    }
```

Modelo Figura

```java
public class Figura {
    private long identificador;
    private String color;
    private double area;
    private double perimetro;

    public Figura(long identificador, String color, double area, double perimetro) {
        this.identificador = identificador;
        this.color = color;
        this.area = area;
        this.perimetro = perimetro;
    }

    public long getIdentificador() {
        return identificador;
    }

    public void setIdentificador(long identificador) {
        this.identificador = identificador;
    }

    public String getColor() {
        return color;
    }

    public void setColor(String color) {
        this.color = color;
    }

    public double getArea() {
        return area;
    }

    public void setArea(double area) {
        this.area = area;
    }

    public double getPerimetro() {
        return perimetro;
    }

    public void setPerimetro(double perimetro) {
        this.perimetro = perimetro;
    }

    @Override
    public String toString() {
        return "Figura{" + "identificador=" + identificador + ", color=" + color + ", area=" + area + ", perimetro=" + perimetro + '}';
    }
```

Modelo Polígonos

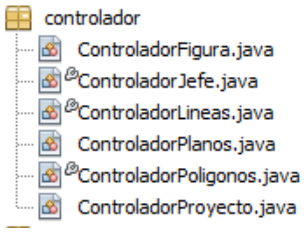
```java
public class Poligonos extends Figura{
    private int numLineas;

    public Poligonos( long identificador, String color, double area, double perimetro) {
        super(identificador, color, area, perimetro); //super indica que datos vienen del
    }

    public Poligonos( long identificador, String color, double area, double perimetro, int numLineas) {
        super(identificador, color, area, perimetro); //super indica que datos vienen del
        this.numLineas = numLineas; //agregamos sueldo bruto a un constructor aparte
    }

    public int getNumLineas() {
        return numLineas;
    }

    public void setNumLineas(int numLineas) {
        this.numLineas = numLineas;
    }

    @Override
    public String toString() {
        return "Poligonos{" + "numLineas=" + numLineas + '}';
    }


}
```

Modelo Líneas

```java
public class Lineas {
    private double puntoOrigen;
    private double puntoFinal;
    private double longitud;
    private long identificador;

    public Lineas(double puntoOrigen, double puntoFinal, double longitud, long identificador) {
        this.puntoOrigen = puntoOrigen;
        this.puntoFinal = puntoFinal;
        this.longitud = longitud;
        this.identificador = identificador;
    }


    public double getPuntoOrigen() {
        return puntoOrigen;
    }

    public void setPuntoOrigen(double puntoOrigen) {
        this.puntoOrigen = puntoOrigen;
    }

    public double getPuntoFinal() {
        return puntoFinal;
    }

    public void setPuntoFinal(double puntoFinal) {
        this.puntoFinal = puntoFinal;
    }

    public double getLongitud() {
        return longitud;
    }

    public void setLongitud(double longitud) {
        this.longitud = longitud;
    }

    public long getIdentificador() {
        return identificador;
    }

    public void setIdentificador(long identificador) {
        this.identificador = identificador;
    }

    @Override
    public String toString() {
        return "Lineas{" + "puntoOrigen=" + puntoOrigen + ", puntoFinal=" + puntoFinal + ", longitud=" + longitud + ", identificador=" + identificador + '}';
    }
```

Controlador

Creamos el controlador. El controlador será el que nos ayude a relacionar la vista con el modelo. Aquí se creara el array list para las listas, se generaran los códigos, y los distintos crear(), buscar(), actualizar(), borrar(), y los getts y setters.



Controlador Jefe

```
package controlador;

import java.util.ArrayList;
import java.util.List;
import modelo.Jefe;

public class ControladorJefe {
    private List<Jefe> listaJefe;
    private Jefe seleccionado;

    public ControladorJefe(){
        listaJefe=new ArrayList<>();
    }

    public long generarCodigo(){
        return(listaJefe.size()>0)? listaJefe.get(listaJefe.size()-1).getCodigo()+1:1;
    }

    public boolean crear(String Nombre, String Apellido, String direccion, String telefono){
        return listaJefe.add(new Jefe(Nombre, Apellido, generarCodigo(), direccion, telefono));
    }

    public Jefe buscar(long codigo){
        for(Jefe jefe:listaJefe){
            if(jefe.getCodigo()==(codigo)){
                seleccionado=jefe;
                return jefe;
            }
        }
        return null;

    }

    public boolean actualizar (String Nombre, String Apellido, long codigo, String direccion, String telefono){
        Jefe jefe = buscar(codigo);
        if (jefe!=null){
            int posicion = listaJefe.indexOf(jefe);
            jefe.setNombre(Nombre);
            jefe.setApellido(Apellido);
            jefe.setDireccion(direccion);
            jefe.setTelefono(telefono);
            listaJefe.set(posicion, jefe);
            return true;
        }
        return false;
    }

    public boolean eliminar(long codigo){
        Jefe jefe=buscar(codigo);
        return listaJefe.remove(jefe);
    }

    public List<Jefe> getListaJefe() {
        return listaJefe;
    }

    public void setListaJefe(List<Jefe> listaJefe) {
        this.listaJefe = listaJefe;
    }

    public Jefe getSeleccionado() {
        return seleccionado;
    }

    public void setSeleccionado(Jefe seleccionado) {
        this.seleccionado = seleccionado;
    }
```

Controlador Proyecto

```java
import java.util.ArrayList;
import java.util.List;
import modelo.Proyecto;

/**
 *
 * @author Usuario
 */
public class ControladorProyecto {
    private List<Proyecto> listaProyecto;
    private Proyecto seleccionado;

    public ControladorProyecto(){
        listaProyecto=new ArrayList<>();
    }
    public long generarCodigo(){
        return(listaProyecto.size()>0)? listaProyecto.get(listaProyecto.size()-1).getCodigo()+1:1;
    }
    public boolean crear(String Nombre){
        return listaProyecto.add(new Proyecto(generarCodigo(),Nombre));
    }

    public Proyecto buscar(long codigo){
       for(Proyecto proyecto:listaProyecto){
            if(proyecto.getCodigo()==(codigo)){
                seleccionado=proyecto;
                return proyecto;
            }
        }
        return null;
    }

    public boolean actualizar(long codigo, String Nombre){
        Proyecto proyecto = buscar(codigo);
        if (proyecto!=null){
            int posicion = listaProyecto.indexOf(proyecto);
            proyecto.setNombre(Nombre);

            return true;
        }
        return false;
    }
    public boolean eliminar(long codigo){
        Proyecto proyecto=buscar(codigo);
        return listaProyecto.remove(proyecto);
    }
```

```java
    public List<Proyecto> getListaProyecto() {
        return listaProyecto;
    }

    public void setListaProyecto(List<Proyecto> listaProyecto) {
        this.listaProyecto = listaProyecto;
    }

    public Proyecto getSeleccionado() {
        return seleccionado;
    }

    public void setSeleccionado(Proyecto seleccionado) {
        this.seleccionado = seleccionado;
    }
```

Controlador Planos

```java
import java.util.ArrayList;
import java.util.List;
import modelo.Planos;

/**
 *
 * @author Usuario
 */
public class ControladorPlanos {
    private List<Planos>listaPlanos;
    private Planos seleccionado;

    public ControladorPlanos(){
        listaPlanos=new ArrayList<>();
    }

    public long generarIdentificacion(){
        return(listaPlanos.size()>0)? listaPlanos.get(listaPlanos.size()-1).getIdentificacion()+1:1;
    }

    public boolean crear(String Fecha, int numArquitectos, int numFiguras){
        return listaPlanos.add(new Planos(generarIdentificacion(),Fecha, numArquitectos, numFiguras));
    }

    public Planos buscar(long Identificacion){
        for(Planos planos:listaPlanos){
            if(planos.getIdentificacion()==(Identificacion)){
                seleccionado=planos;
                return planos;
            }
        }
        return null;
    }

    public boolean actualizar(long Identificacion, String Fecha, int numArquitectos, int numFiguras){
        Planos planos = buscar(Identificacion);
        if (planos!=null){
            int posicion = listaPlanos.indexOf(planos);
            planos.setFecha(Fecha);
            planos.setNumArquitectos(numArquitectos);
            planos.setNumFiguras(numFiguras);
            listaPlanos.set(posicion, planos);
            return true;
        }
        return false;
    }

    public boolean eliminar(long Identificacion){
        Planos planos=buscar(Identificacion);
        return listaPlanos.remove(planos);
    }

    public List<Planos> getListaPlanos() {
        return listaPlanos;
    }

    public void setListaPlanos(List<Planos> listaPlanos) {
        this.listaPlanos = listaPlanos;
    }

    public Planos getSeleccionado() {
        return seleccionado;
    }

    public void setSeleccionado(Planos seleccionado) {
        this.seleccionado = seleccionado;
    }
}
```

Controlador Figura

```java
import java.util.ArrayList;
import java.util.List;
import modelo.Figura;

/**
 *
 * @author Usuario
 */
public class ControladorFigura {
    private List<Figura> listaFigura;
    private Figura seleccionado;

    public ControladorFigura(){
        listaFigura=new ArrayList<>();
    }

    public long generarIdentificador(){
        return(listaFigura.size()>0)? listaFigura.get(listaFigura.size()-1).getIdentificador()+1:1;
    }

    public boolean crear(String color, double area, double perimetro){
        return listaFigura.add(new Figura(generarIdentificador(),color, area, perimetro));
    }

    public Figura buscar(long Identificador){
        for(Figura figura:listaFigura){
            if(figura.getIdentificador()==(Identificador)){
                seleccionado=figura;
                return figura;
            }
        }
        return null;
    }

    public boolean actualizar(long Identificador, String color, double area, double perimetro){
        Figura figura = buscar(Identificador);
        if (figura!=null){
            int posicion = listaFigura.indexOf(figura);
            figura.setColor(color);
            figura.setArea(area);
            figura.setPerimetro(perimetro);
            listaFigura.set(posicion, figura);
            return true;
        }
        return false;
    }

    public boolean eliminar(long Identificador){
        Figura figura=buscar(Identificador);
        return listaFigura.remove(figura);
    }

    public List<Figura> getListaFigura() {
        return listaFigura;
    }

    public void setListaFigura(List<Figura> listaFigura) {
        this.listaFigura = listaFigura;
    }

    public Figura getSeleccionado() {
        return seleccionado;
    }

    public void setSeleccionado(Figura seleccionado) {
        this.seleccionado = seleccionado;
    }


}
```

Controlador Polígonos

```java
import java.util.ArrayList;
import java.util.List;
import modelo.Figura;
import modelo.Poligonos;

public class ControladorPoligonos {
    private List<Poligonos> listaPoligonos;
    private Poligonos seleccionado;

    public ControladorPoligonos(){
        listaPoligonos=new ArrayList<>();
        seleccionado=null;
    }

    public boolean crear(long identificador, String color, double area, double perimetro, int numLineas){
        Poligonos e=new Poligonos(identificador, color, area, perimetro, numLineas);
        return listaPoligonos.add(e);
    }

    public Poligonos buscar(long identificador){
        for (Poligonos poligonos : listaPoligonos) {
            if(poligonos.getIdentificador()==(identificador))
                return poligonos;
        }
        return null;
    }

    public boolean actualizar(long identificador, String color, double area, double perimetro, int numLineas){
        Poligonos poligonos=this.buscar(identificador);
        if(poligonos!=null){
            int posicion=listaPoligonos.indexOf(poligonos);
            poligonos.setColor(color);
            poligonos.setArea(area);
            poligonos.setPerimetro(perimetro);
            poligonos.setNumLineas(numLineas);
            listaPoligonos.set(posicion, poligonos);
            return true;
        }
        return false;
    }

    public boolean eliminar(long identificador){
        Poligonos poligonos=this.buscar(identificador);
        if(poligonos!=null){
            return listaPoligonos.remove(poligonos);
        }
        return false;
    }

        public List<Poligonos> getListaPoligonos() {
            return listaPoligonos;
        }

        public void setListaPoligonos(List<Poligonos> listaPoligonos) {
            this.listaPoligonos = listaPoligonos;
        }

        public Poligonos getSeleccionado() {
            return seleccionado;
        }

        public void setSeleccionado(Poligonos seleccionado) {
            this.seleccionado = seleccionado;
        }

    }
```

Controlador Líneas

```java
import java.util.ArrayList;
import java.util.List;
import modelo.Lineas;

public class ControladorLineas {
    private List<Lineas> listaLineas;
    private Lineas seleccionado;

    public ControladorLineas(){
        listaLineas=new ArrayList<>();
    }

    public long generarIdentificador(){
        return(listaLineas.size()>0)? listaLineas.get(listaLineas.size()-1).getIdentificador()+1:1;
    }

    public boolean crear(double puntoOrigen, double puntoFinal, double longitud){
        return listaLineas.add(new Lineas(puntoOrigen, puntoFinal, longitud, generarIdentificador()));
    }

    public Lineas buscar(long identificador){
       for(Lineas lineas:listaLineas){
            if(lineas.getIdentificador()==(identificador)){
                seleccionado=lineas;
                return lineas;
            }
        }
        return null;
    }

    public boolean actualizar (double puntoOrigen, double puntoFinal, double longitud, long identificador){
        Lineas lineas = buscar(identificador);
        if (lineas!=null){
            int posicion = listaLineas.indexOf(lineas);
            lineas.setPuntoOrigen(puntoOrigen);
            lineas.setPuntoFinal(puntoFinal);
            lineas.setLongitud(longitud);
            listaLineas.set(posicion, lineas);
            return true;
        }
        return false;
    }

    public boolean eliminar(long identificador){
        Lineas lineas=buscar(identificador);
        return listaLineas.remove(lineas);
    }
```

```java
    public List<Lineas> getListaLineas() {
        return listaLineas;
    }

    public void setListaLineas(List<Lineas> listaLineas) {
        this.listaLineas = listaLineas;
    }

    public Lineas getSeleccionado() {
        return seleccionado;
    }

    public void setSeleccionado(Lineas seleccionado) {
        this.seleccionado = seleccionado;
    }

}
```

Vista

Se crea la vista y en este irán los menos de interacción con el usuario, de cada una de las secciones, se mostrará una vista general para elegir las diferentes opciones.

Vista Jefe

```java
package vista;

import controlador.ControladorJefe;
import java.util.Scanner;
import modelo.Jefe;

public class vistaJefe {
    public Scanner teclado;
    private ControladorJefe controladorJefe;
    public vistaJefe(ControladorJefe controladorJefe){
        teclado=new Scanner(System.in);
        this.controladorJefe=controladorJefe;
    }

    public void menu(){
        int opcion=0;
        do{
            System.out.println("1. Crear \n 2. Actualizar \n 3. Buscar \n 4. Eliminar \n 5. Listar \n 6. Sair");
            opcion=teclado.nextInt();
            switch(opcion){
                case 1:this.crear(); break;
                case 2:this.actualizar(); break;
                case 3:this.buscar(); break;
                case 4:this.eliminar(); break;
                case 5:this.listar(); break;
            }
        }while(true);
    }

    public void crear()
    {
        System.out.println("Ingrese: \n Nombre");
        String Nombre=teclado.next();
        System.out.println("Apellido");
        String Apellido=teclado.next();
        System.out.println("Direccion");
        String Direccion=teclado.next();
        System.out.println("Telefono");
        String telefono=teclado.next();
        System.out.println("Res: "+ controladorJefe.crear(Nombre,Apellido, Direccion, telefono));
    }
```

```
    public void actualizar(){
        System.out.println("Ingrese: \n Nombre");
        String Nombre=teclado.next();
        System.out.println("Ingrese: \n Apellido");
        String Apellido=teclado.next();
        System.out.println("Ingrese: \n Codigo");
        long codigo=teclado.nextLong();
        System.out.println("Ingrese: \n Direccion");
        String Direccion=teclado.next();
        System.out.println("Ingrese: \n Telefono");
        String telefono=teclado.next();
        System.out.println("Res: "+controladorJefe.actualizar(Nombre, Apellido, codigo, Direccion,  telefono));
    }

    public void buscar(){
        System.out.println("Ingrese: \n Codigo");
        long codigo=teclado.nextLong();
        System.out.println(controladorJefe.buscar(codigo));
    }

    public void eliminar(){
        this.buscar();
        System.out.println("Res: "+controladorJefe.eliminar(controladorJefe.getSeleccionado().getCodigo()));
    }

    public void listar(){
        for(Jefe jefe : controladorJefe.getListaJefe()){
            System.out.println(jefe);
        }
    }
}
```

Vista Proyecto

```
package vista;

import controlador.ControladorProyecto;
import java.util.Scanner;
import modelo.Proyecto;

public class vistaProyecto {
    public Scanner teclado;
    private ControladorProyecto controladorProyecto;

    public vistaProyecto(ControladorProyecto controladorProyecto){
        teclado=new Scanner(System.in);
        this.controladorProyecto=controladorProyecto;
    }

    public void menu(){
        int opcion=0;
        do{
            System.out.println("1. Crear \n 2. Actualizar \n 3. Buscar \n 4. Eliminar \n 5. Listar \n 6. Sair");
            opcion=teclado.nextInt();
            switch(opcion){
                case 1:this.crear(); break;
                case 2:this.actualizar(); break;
                case 3:this.buscar(); break;
                case 4:this.eliminar(); break;
                case 5:this.listar(); break;
            }
        }while(true);
    }

    public void crear()
    {
        System.out.println("Ingrese: \n Nombre");
        String Nombre=teclado.next();
        System.out.println("Res: "+ controladorProyecto.crear(Nombre));
    }

    public void actualizar(){
        System.out.println("Ingrese: \n Codigo");
        long codigo=teclado.nextLong();
        System.out.println("Ingrese: \n Nombre");
        String Nombre=teclado.next();
        System.out.println("Res: "+controladorProyecto.actualizar(codigo, Nombre));
    }
```

```
    public void buscar(){
        System.out.println("Ingrese: \n Codigo");
        long codigo=teclado.nextLong();
        System.out.println(controladorProyecto.buscar(codigo));
    }

    public void eliminar(){
        this.buscar();
        System.out.println("Res: "+controladorProyecto.eliminar(controladorProyecto.getSeleccionado().getCodigo()));
    }

    public void listar(){
        for(Proyecto proyecto : controladorProyecto.getListaProyecto()){
            System.out.println(proyecto);
        }
    }


}
```

Vista Planos

```
package vista;

import controlador.ControladorPlanos;
import java.util.Scanner;
import modelo.Planos;

public class VistaPlanos {
    public Scanner teclado;
    private ControladorPlanos controladorPlanos;

    public VistaPlanos(ControladorPlanos controladorPlanos){
        teclado=new Scanner(System.in);
        this.controladorPlanos=controladorPlanos;
    }

    public void menu(){
        int opcion=0;
        do{
            System.out.println("1. Crear \n 2. Actualizar \n 3. Buscar \n 4. Eliminar \n 5. Listar \n 6. Sair");
            opcion=teclado.nextInt();
            switch(opcion){
                case 1:this.crear(); break;
                case 2:this.actualizar(); break;
                case 3:this.buscar(); break;
                case 4:this.eliminar(); break;
                case 5:this.listar(); break;
            }
        }while(true);
    }

    public void crear()
    {
        System.out.println("Ingrese: \n Fecha");
        String Fecha=teclado.next();
        System.out.println("Numero de Arquitectos");
        int numArqui=teclado.nextInt();
        System.out.println("Numero de Figuras");
        int numFiguras=teclado.nextInt();
        System.out.println("Res: "+ controladorPlanos.crear(Fecha, numArqui, numFiguras));
    }
```

```
    public void actualizar(){
        System.out.println("Ingrese: \n Identificacion");
        long identificacion=teclado.nextLong();
        System.out.println("Ingrese: \n Fecha");
        String fecha=teclado.next();
        System.out.println("Ingrese: \n Numero de Arquitectos");
        int numArqui=teclado.nextInt();
        System.out.println("Ingrese: \n Numero de Figuras");
        int numFiguras=teclado.nextInt();
        System.out.println("Res: "+controladorPlanos.actualizar(identificacion, fecha, numArqui, numFiguras));
    }

    public void buscar(){
        System.out.println("Ingrese: \n identificacion de los planos");
        long identificacion=teclado.nextLong();
        System.out.println(controladorPlanos.buscar(identificacion));
    }

    public void eliminar(){
        this.buscar();
        System.out.println("Res: "+controladorPlanos.eliminar(controladorPlanos.getSeleccionado().getIdentificacion()));
```

Vista Figura

```
import controlador.ControladorFigura;
import java.util.Scanner;
import modelo.Figura;

public class vistaFigura {
    public Scanner teclado;
    private ControladorFigura controladorFigura;
    public vistaFigura(ControladorFigura controladorFigura){
        teclado=new Scanner(System.in);
        this.controladorFigura=controladorFigura;
    }

    public void menu(){
        int opcion=0;
        do{
            System.out.println("1. Crear \n 2. Actualizar \n 3. Buscar \n 4. Eliminar \n 5. Listar \n 6. Sair");
            opcion=teclado.nextInt();
            switch(opcion){
                case 1:this.crear(); break;
                case 2:this.actualizar(); break;
                case 3:this.buscar(); break;
                case 4:this.eliminar(); break;
                case 5:this.listar(); break;
            }
        }while(true);
    }

    public void crear()
    {
        System.out.println("Ingrese: \n Color");
        String color=teclado.next();
        System.out.println("Ingrese: \n Area");
        double area=teclado.nextDouble();
        System.out.println("Ingrese: \n Perimetro");
        double perimetro=teclado.nextDouble();
        System.out.println("Res: "+ controladorFigura.crear(color, area, perimetro));
    }

    public void actualizar(){
        System.out.println("Ingrese: \n Indentificador");
        long identificador=teclado.nextLong();
        System.out.println("Ingrese: \n Color");
        String color=teclado.next();
        System.out.println("Ingrese: \n Area");
        double area=teclado.nextDouble();
        System.out.println("Ingrese: \n Perimetro");
        double perimetro=teclado.nextDouble();
        System.out.println("Res: "+controladorFigura.actualizar(identificador, color, area, perimetro));
    }


    public void buscar(){
        System.out.println("Ingrese: \n Codigo");
        long codigo=teclado.nextLong();
        System.out.println(controladorFigura.buscar(codigo));
    }

    public void eliminar(){
        this.buscar();
        System.out.println("Res: "+controladorFigura.eliminar(controladorFigura.getSeleccionado().getIdentificador()));
    }

    public void listar(){
        for(Figura figura : controladorFigura.getListaFigura()){
            System.out.println(figura);
        }
    }
}
```

Vista Polígonos

```java
import controlador.ControladorPoligonos;
import java.util.Scanner;
import modelo.Poligonos;
import vista.vistaLineas;
import controlador.ControladorLineas;

public class vistaPoligonos {
    private ControladorPoligonos controladorPoligonos;
    private final Scanner teclado;
    private ControladorLineas controladorLineas;

    public vistaPoligonos(){
        controladorPoligonos=new ControladorPoligonos();
        teclado=new Scanner(System.in);
    }

    public void menu(){
        int opcion=0;
        do{
            System.out.println("1. Crear \n 2. Actualizar \n 3. Buscar \n 4. Eliminar \n 5. Listar \n 6. Sair");
            opcion=teclado.nextInt();
            switch(opcion){
                case 1:this.crear(); break;
                case 2:this.actualizar(); break;
                case 3:this.buscar(); break;
                case 4:this.eliminar(); break;
                case 5:this.listar(); break;
            }
        }while(true);
    }

    public double perimetroPoligono(int numLineas)
    {
        double lado;
        double perimetro = 0;
        for(int i = 1; i <= numLineas; i++)
        {
            System.out.println("Ingrese el lado " + i + ": ");
            System.out.println("Ingrese: \n Punto de Origen");
            double puntoOrigen=teclado.nextDouble();
            System.out.println("Ingrese: \n Punto del Final");
            double puntoFinal=teclado.nextDouble();
            double longitud=puntoFinal-puntoOrigen;

            perimetro = perimetro + longitud;
        }
        return perimetro;
    }

    public void crear(){
        System.out.println("Ingresar identificador");
        long identificador=teclado.nextLong();
        System.out.println("Ingresar color");
        String color=teclado.next();
        System.out.println("Ingresar Numero de lineas");
        int numLineas=teclado.nextInt();
        System.out.println("Ingresar area");
        double area=teclado.nextDouble();
        double perimetro=perimetroPoligono(numLineas);
        System.out.println("El perimetro es: " + perimetro);
        System.out.println("Resultado "+controladorPoligonos.crear(identificador,color, area, perimetro, numLineas));

    }

    public void actualizar(){
        System.out.println("Ingrese: \n Identificador");
        long identificador=teclado.nextLong();
        System.out.println("Ingrese: \n Color");
        String color=teclado.next();
        System.out.println("Ingrese: \n Area");
        double area=teclado.nextDouble();
        System.out.println("Ingrese: \n Perimetro");
        double perimetro=teclado.nextDouble();
        System.out.println("Numero de Lineas");
        int numLineas=teclado.nextInt();
        System.out.println("Res: "+controladorPoligonos.actualizar(identificador, color, area, perimetro, numLineas));
    }

    public void buscar(){
        System.out.println("Ingrese: \n Identificador");
        long identificador=teclado.nextLong();
        System.out.println(controladorPoligonos.buscar(identificador));
    }

    public void eliminar(){
        this.buscar();
        System.out.println("Res: "+controladorPoligonos.eliminar(controladorPoligonos.getSeleccionado().getIdentificador()));
    }

    public void listar(){
        for(Poligonos poligonos : controladorPoligonos.getListaPoligonos()){
            System.out.println(poligonos);
        }
    }
}
```

Vista Líneas

```java
import java.util.Scanner;
import controlador.ControladorLineas;
import modelo.Lineas;

public class vistaLineas {
    public Scanner teclado;
    private ControladorLineas controladorLineas;

    public vistaLineas(ControladorLineas controladorLineas){
        teclado=new Scanner(System.in);
        this.controladorLineas=controladorLineas;
    }

    public void menu(){
        int opcion=0;
        do{
            System.out.println("1. Crear \n 2. Actualizar \n 3. Buscar \n 4. Eliminar \n 5. Listar \n 6. Sair");
            opcion=teclado.nextInt();
            switch(opcion){
                case 1:this.crear(); break;
                case 2:this.actualizar(); break;
                case 3:this.buscar(); break;
                case 4:this.eliminar(); break;
                case 5:this.listar(); break;
            }
        }while(true);
    }

    public double crear()
    {
        System.out.println("Ingrese: \n Punto de Origen");
        double puntoOrigen=teclado.nextDouble();
        System.out.println("Ingrese: \n Punto del Final");
        double puntoFinal=teclado.nextDouble();
        double longitud=puntoFinal-puntoOrigen;
        System.out.println("La longitud es: " + longitud);
        System.out.println("Res: "+ controladorLineas.crear(puntoOrigen, puntoFinal, longitud));
        return longitud;
    }

    public void actualizar(){
        System.out.println("Ingrese: \n Identificador");
        long identificador=teclado.nextLong();
        System.out.println("Ingrese: \n punto de Origen");
        double puntoOrigen=teclado.nextDouble();
        System.out.println("Ingrese: \n Punto del final");
        double puntoFinal=teclado.nextDouble();;
        System.out.println("Ingrese: \n Longitud");
        double longitud=teclado.nextDouble();
        System.out.println("Res: "+controladorLineas.actualizar(puntoOrigen, puntoFinal, longitud, identificador ));
    }

    public void buscar(){
        System.out.println("Ingrese: \n identificador de las Lineas");
        long identificador=teclado.nextLong();
        System.out.println(controladorLineas.buscar(identificador));
    }

    public void eliminar(){
        this.buscar();
        System.out.println("Res: "+controladorLineas.eliminar(controladorLineas.getSeleccionado().getIdentificador()));
    }

    public void listar(){
        for(Lineas lineas : controladorLineas.getListaLineas()){
            System.out.println(lineas);
        }
    }
}
```

Vista General

Aquí llamamos a todos los métodos desde esta clase.

```java
package vista;

import java.util.Scanner;
import controlador.ControladorFigura;
import controlador.ControladorJefe;
import controlador.ControladorLineas;
import controlador.ControladorPlanos;
import controlador.ControladorPoligonos;
import controlador.ControladorProyecto;

public class vistaGeneral {
    public Scanner teclado;
    public VistaPlanos vistaPlanos;
    public ControladorPlanos controladorPlanos;
    public vistaFigura vistaFigura;
    public ControladorFigura controladorFigura;
    public vistaJefe vistaJefe;
    public ControladorJefe controladorJefe;
    public vistaLineas vistaLineas;
    public ControladorLineas controladorLineas;
    public vistaPoligonos vistaPoligonos;
    public ControladorPoligonos controladorPoligonos;
    public vistaProyecto vistaProyecto;
    public ControladorProyecto controladorProyecto;

    public vistaGeneral(){
        teclado = new Scanner(System.in);
        controladorPlanos=new ControladorPlanos();
        controladorFigura=new ControladorFigura();
        controladorJefe=new ControladorJefe();
        controladorLineas=new ControladorLineas();
        controladorPoligonos=new ControladorPoligonos();
        controladorProyecto=new ControladorProyecto();
        vistaPoligonos=new vistaPoligonos();
        vistaPlanos=new VistaPlanos(controladorPlanos);
        vistaFigura=new vistaFigura(controladorFigura);
        vistaJefe=new vistaJefe(controladorJefe);
        vistaLineas=new vistaLineas(controladorLineas);

        vistaProyecto=new vistaProyecto(controladorProyecto);
    }
```

```java
    public void menu(){
        int op=0;
        do{
            System.out.println("Seleccionar una opción");
            System.out.println("1.Jefe de Proyecto");
            System.out.println("2.Proyecto");
            System.out.println("3.Planos");
            System.out.println("4.Figura");
            System.out.println("5.Poligonos");
            System.out.println("6.Lineas");
            op=teclado.nextInt();
            switch(op){
                case 1:vistaJefe.menu();break;
                case 2:vistaProyecto.menu();break;
                case 3:vistaPlanos.menu();break;
                case 4:vistaFigura.menu();break;
                case 5:vistaPoligonos.menu();break;
                case 6:vistaLineas.menu();break;
            }
        }while(op<7);
    }
}
```

Se manda a mostrar al correr el programa de la siguiente manera

```
Seleccionar una opción
1.Jefe de Proyecto
2.Proyecto
3.Planos
4.Figura
5.Poligonos
6.Lineas
5
1. Crear
 2. Actualizar
 3. Buscar
 4. Eliminar
 5. Listar
 6. Sair
1

Ingresar identificador
8904
Ingresar color
rojo
Ingresar Numero de lineas
3
Ingresar area
6
Ingrese el lado 1:
Ingrese:
 Punto de Origen
3
Ingrese:
 Punto del Final
5

 Ingrese el lado 2:
 Ingrese:
  Punto de Origen
 4
 Ingrese:
  Punto del Final
 6
 Ingrese el lado 3:
 Ingrese:
  Punto de Origen
 5
 Ingrese:
  Punto del Final
 7
 El perimetro es: 6.0
```